

MEMORANDUM
RM-5573-ARPA
AUGUST 1968

AD 675040

THE ON-LINE FIRING SQUAD SIMULATOR

R. M. Balzer and R. W. Shirey

PREPARED FOR:
ADVANCED RESEARCH PROJECTS AGENCY

DDC
SEP 27 1968

The **RAND** *Corporation*
SANTA MONICA • CALIFORNIA

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va 22151

38

MEMORANDUM
RM-5573-ARPA
AUGUST 1968

THE ON-LINE
FIRING SQUAD SIMULATOR

R. M. Balzer and R. W. Shirey

This research is supported by the Advanced Research Projects Agency under Contract No. DAH15 67 C 0111. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of ARPA.

DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

PREFACE

This Memorandum describes an on-line, graphical, man/machine interactive computer system that employs the RAND Tablet. The system provides aids for solving a particular problem of interactive design, and, it is hoped, will result in significant new solutions. The firing squad synchronization problem was chosen because it affords an opportunity for evaluating the problem-solving power of both the particular aids and such systems as a whole.

R. W. Shirey participated in the 1967 RAND Summer Graduate Program and is now a RAND Consultant. He is presently a doctoral candidate in Computer Sciences at the University of Wisconsin.

SUMMARY

This Memorandum describes a computer system designed both to investigate man/machine graphical communications and to find improved solutions for the firing squad synchronization problem. The system provides aids that allow the user to approach this problem by methods he might otherwise not attempt because of the tedious hand calculations required. Furthermore, the graphical nature of the system and the type of aids provided combine to influence significantly the attitude of the experimenter toward various solution approaches.

First, the authors state the problem and note some of its inherent difficulties. Next, they discuss the necessary tasks for solving the problem, and then go on to show how and why some of these tasks should be automated. Then, finally, the authors discuss general principles learned while building the system, and make recommendations concerning the cost and advisability of constructing similar systems.

CONTENTS

PREFACE	iii
SUMMARY	v
Section	
I. INTRODUCTION	1
II. PROBLEM STATEMENT	3
III. COMMON APPROACHES AND BASIC CONSIDERATIONS .	6
Solution by Hand	7
Solving with Computer Aids	9
Summary Remarks on the Problem	10
IV. THE SYSTEM IN GENERAL	11
Hardware, Software and Interaction	11
Number of States and External State Names	14
The Message Center	14
Entry, Storage and Retrieval of Function	
Values	15
Simple Simulation of a Firing Squad	16
Enter Constraints	19
Simulation with Constraints	20
Simulation with Backtracking	21
Frozen and Free Productions	24
Snap View and Bright Positions	24
History Scroll, Freezing and Deletion	25
Image Solutions	26
Miscellaneous	29
V. IMPLEMENTATION EXPERIENCE AND	
PREREQUISITES	30
VI. CONCLUSION	32
REFERENCES	33

I. INTRODUCTION

The purpose of this study is to investigate man/machine interaction in the context of solving a conceptually difficult, formal problem. We want a problem that requires no specialized knowledge, so that a fair comparison can be made between computer-aided and unaided attempts at solution. We also want a problem that is graphic. The firing squad synchronization problem satisfies these criteria extremely well. It has the added advantage that no optimal solution has yet been produced.

The system designed for these purposes is essentially a collection of problem-solving aids that can be divided into three main groups: the first includes bookkeeping aids, useful displays of information, ability to get hard copy, and other basic services; the second, means for testing and simulating solutions; the third, specialized, high-level heuristic aids for creating solutions. All three groups attempt to extend the user's power in exploring the universe of the problem, enabling and encouraging him to approach the problem in ways that might otherwise be prohibited by immense amounts of necessary hand calculations or the human tendency toward error.

We hope that this system will result in interesting new solutions to the firing squad problem, and will provide new information on the reactions of humans in such man/machine interactive environments.

We begin by stating the problem and noting some of its inherent difficulties. Next, we discuss the necessary tasks for solving the problem, and then go on to show how and why some of these tasks should be automated. Then, finally, we make general recommendations concerning the design of similar computer systems, based on the experience gained while constructing this one.

II. PROBLEM STATEMENT

This Memorandum concerns a problem publically first presented in 1964 by E. F. Moore [1]:

The problem known as the firing squad synchronization problem was devised about the year 1957 by John Myhill, but so far as I know the statement of the problem has not yet appeared in print. It has been widely circulated by word of mouth, and has attracted sufficient interest that it ought to be available in print. The problem first arose in connection with causing all parts of a self-reproducing machine to be turned on simultaneously. The problem was first solved by John McCarthy and Marvin Minsky, and now that it is known to have a solution, even persons with no background in logical design or computer programming can usually find a solution in a time of two to four hours. The problem has an unusual elegance in that it is directly analogous to problems of logical design, systems design, or programming, but it does not depend on the properties of any particular set of logical elements or the instructions of any particular computer. I would urge those who know a solution to this problem to avoid divulging it to those who are figuring it out for themselves, since this will spoil the fun of this intriguing problem.

Consider a finite (but arbitrarily long) one dimensional array of finite-state machines, all of which are alike except the ones at each end. The machines are called soldiers, and one of the end machines is called a general. The machines are synchronous, and the state of each machine at time $t + 1$ depends on the states of itself and of its two neighbors at time t . The problem is to specify the states and transitions of the soldiers in such a way that the general can cause them to go into one particular terminal state (i.e., they fire their guns) all at exactly the same time. At the beginning (i.e., $t = 0$) all the soldiers are assumed to be in a single state, the quiescent state. When the general undergoes the transition into the state labeled "Fire when ready," he does not take any initiative afterwards, and the rest is up to the soldiers. The signal can propagate down the line no faster than one soldier per unit of time, and their problem is how to get all coordinated and in rhythm. The tricky part of the problem is that the same kind of soldier with a fixed number K of states is required to be able to do this, regardless of the length N of the firing squad.

In particular, the soldier with K states should work correctly, even when N is much larger than K . Roughly speaking, none of the soldiers is permitted to count as high as N .

Two of the soldiers, the general and the soldier farthest from the general, are allowed to be slightly different from the other soldiers in being able to act without having soldiers on both sides of them, but their structure must also be independent of N .

A convenient way of indicating a solution of this problem is to use a piece of graph paper, with the horizontal coordinate representing the spatial position and the vertical coordinate representing time. Within the (i, j) square of the graph paper a symbol may be written, indicating the state of the i th soldier at time j . Visual examination of the pattern of propagation of these symbols can indicate what kinds of signaling must take place between the soldiers.

Any solution to the firing squad synchronization problem can easily be shown to require that the time from the general's order until the guns go off must be at least $2N-2$, where N is the number of soldiers. Most persons solve this problem in a way which requires between $3N$ and $8N$ units of time, although occasionally other solutions are found. Some such other solutions require $5/2N$ and of the order of N -squared units of time, for instance. Until recently, it was not known what the smallest possible time for a solution was. However, this was solved at M.I.T. by Professor E. Goto of the University of Tokyo. The solution obtained by Goto used a very ingenious construction, with each soldier having many thousands of states, and the solution required exactly $2N-2$ units of time. In view of the difficulty of obtaining this solution, a much more interesting problem for beginners is to try to obtain some solution between $3N$ and $8N$ units of time, which as remarked above, is relatively easy to do.*

Goto's solution] apparently has not been published.

However, Abraham Waksman [3] has found a 16-state minimal-time solution using essentially the same ideas presented in

*Ref. 1, pp. 213-219.

Sec. II below. P. C. Fischer [4] has also used these ideas in discussing other properties of one-dimensional iterative arrays of finite-state machines. The best solution to date is R. M. Balzer's [5] 8-state minimal-time solution.

III. COMMON APPROACHES AND BASIC CONSIDERATIONS

The firing squad synchronization problem can be solved by successively subdividing the line into any number of equal parts, then subdividing each of these parts similarly, and so on, until all the members of the line become division points, at which time they all fire. Most existing solutions use this technique, and it can provide solutions of minimal time, $2N-2$. Balzer's solution [5] divides the line into halves, quarters, eighths, etc.

Finding a solution entails construction of a finite-state machine by defining for the machine a transition function that yields appropriate behavior when placed in the iterative array. Although automata are usually defined by state tables, here it is easier to interpret a function as a set of rules called productions. These rules take the form

$$\text{LMR} \rightarrow \text{S}.$$

This rule states that if, at time t , a machine is in state M , and the machine on its left is in state L , and the machine on its right is in state R , then the machine's state at time $t+1$ is S . We call S the "resultant" of the production.

In particular, we are concerned only with minimal-time solutions. To treat the problems resulting from the soldiers at each end of the line, we use an additional state

as an end marker, and, at each end of the line, a virtual additional machine which forever remains in the marker state. Since no other machine is ever in the marker state, a single set of productions can be defined for all machines in the array.

Exhaustive search for the function is out of the question, even with the help of a computer, because the number of possible state tables is far too large. For example, if we seek a solution with ten states (plus the end marker), there will be $9^3 + 2 \cdot 9^2 - 2 = 889$ productions. (The problem statement excludes certain productions and fixes the resultant of two others.) Each of the 889 productions can assume ten values, for a total of 10^{889} functions.

SOLUTION BY HAND

While building a function, say with ten states, the experimenter faces a number of separate tasks--some routine, some challenging, many time-consuming and tedious. He obviously must maintain a large production table. Given some table, perhaps only partially completed, he will need to test it on firing squads of different lengths. This simply involves retrieving values from the table and copying them onto graph paper. Both tasks are routine; nevertheless, performing them will consume much of the experimenter's time.

After several attempts, he may discover that some productions are more important than others, that they are keys to the solution, and he might wish to mark these in order to

remind himself that their values should not be altered without special consideration.

The challenging tasks are the creative ones, and the foremost of these is the creation of ingenious approaches to the problem. These schemes usually appear as a two-dimensional plan for propagation of signals along the squad through time. One method for simultaneously implementing and testing an approach is to draw on the graph paper a skeleton diagram of the intended function behavior, and then force the productions to conform to this plan. This method of defining productions eliminates many false steps.

Special cases arise when the squad is quite short, say less than fifteen men. After a large portion of the production set is defined, especially key productions, and the function has been tested on longer squads, exhaustive search may become feasible for filling in the special productions required for these cases.

If an error occurs in a simulation, such as a soldier firing too early or too late, or if contradictions arise while attempting to fit productions to a behavior skeleton, some production must be changed. The experimenter then becomes interested in why he originally made this definition. Therefore he finds it useful to keep a history of production usage, particularly a table of first usages in the simulation he is currently considering.

In all these tasks there is a high probability of human error due to the large size of the tables, the large number of separate acts to be performed, and, of course, the repetitious nature of most of the work.

SOLVING WITH COMPUTER AIDS

The mechanically repetitious nature of some tasks naturally leads to thoughts of automating them--providing computer aids for the experimenter. The obvious candidates for automation are those tasks which primarily consist of information storage and retrieval, such as table maintenance and simulation. Exhaustive search, where feasible, is handled best by a computer. Having provided these basic services, other more sophisticated tools become possible as well. Finally, the graphic nature of both the problem and the methods previously described influences the choice of computing hardware; graphic input and output quickly come to mind.

The use of interactive graphic equipment is implied because the reactions of humans to a computing system are highly important. A rapid interaction between man and machine tends to stimulate the intuition and perceptivity of the experimenter; immediate response from the machine maintains a high level of human cerebral activity. Just as not having computer aids at all, using them off-line would slow the response from a second or less to hours. Progress might become so slow that the user would lose interest in the problem.

SUMMARY REMARKS ON THE PROBLEM

Let us summarize the above discussion--of the problem and the comparison between attempts at solution with and without computer aids--in order to draw some conclusions about the value of this study.

The problem is interesting enough to have attracted wide attention, but difficult enough that no optimal solution has been demonstrated. It requires no special background, and is simple enough that at least an inefficient solution can be found by hand in a few hours. Conversely, it is rich enough to suggest a computer implementation of a number of tools and techniques to aid the investigator. Also, it is naturally oriented toward the use of interactive graphic hardware.

Furthermore, since exhaustive search for a solution is not practical, the computer aids are only tools, and the user still must provide the creative insights and approaches necessary to finding a solution. Thus, the firing squad synchronization problem is a particularly suitable vehicle for evaluating the effectiveness of interactive, graphical problem-solving aids by comparing their effects with the results of unaided efforts.

IV. THE SYSTEM IN GENERAL

The Firing Squad Synchronization, Simulation and Solution System (FS5) is a highly interactive, graphical computer system. It furnishes three basic groups of tools: the first includes bookkeeping for tables; the second deals with simulation and testing; a third contains the more sophisticated tools, including the ability to draw and implement a skeleton plan, request exhaustive searches, and other functions not obviously needed, but included on the basis of experience with the problem. Associated with these three main categories is a corona of minor devices (e.g., for obtaining hard copy of displays).

HARDWARE, SOFTWARE AND INTERACTION

The FS5 program is written in IBM System/360 PL/I language and runs on an IBM System/360 Model 40. A user communicates with the computer via a RAND Tablet [6] in conjunction with an IBM 2250 cathode ray tube (CRT) display. The tablet hardware consists of a horizontal 10.24-inch-square writing surface and a pen-like writing instrument, together having a resolution of 100 lines per inch along both Cartesian coordinates.

As the user moves the stylus near the tablet surface, a (hardware generated) dot on the CRT follows the stylus motion; this direct feedback helps the user to position the stylus for pointing or drawing. When he presses the stylus against the tablet writing surface, a switch in the

stylus closes, notifying the computer that the user is beginning a stroke. As he moves the stylus across the tablet, the stylus track is displayed (via software) on the CRT; the stylus thus seems to have "ink." When the stylus is lifted, its switch is opened notifying the computer of a stroke completion, and "inking" ceases. A user may "point" at an area on the CRT by closing and opening the stylus switch on the corresponding area of the tablet surface.

The FS5 program uses a set of graphics subroutines written at RAND and called the Integrated Graphics System (IGS). Both character and geometric pattern recognition are included in IGS [7]. A character written by the user is replaced on the display by the corresponding machine-generated character.

The FS5 system presents the user with a picture of a control panel (Fig. 1). The controls are used as if they were physical buttons; they are "pushed" by touching them with the stylus. Problem information is displayed in three main areas. On the left, FS5 shows the simulations of firing squads from length one to length 25. On the right, there is a scroll display of production-usage history. At the top center, FS5 offers a variety of messages concerning its own use and status. The use and function of the controls are described in the following sections.

0			MESSAGE CENTER			
1					SCROLL	
2					UP DOWN	
3						
4						
5						
6						
7	LENGTH 25		# OF STATES		20	
8	STATES		A B C D E F G H I J K L M N O P Q R S T			
9	IMAGES		A B C D E F G H I J K L M N O P Q R S T			
10						
11						
12						
13			FIRST OCCURENCE RESPONSES			
14			STOP AUTO NORMAL BRIGHT			
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						
39						
40						
41						
42						
43						
44						
45						
46						
47						
48						

Fig. 1--The Firing Squad Simulator Scope Face

NUMBER OF STATES AND EXTERNAL STATE NAMES

Suppose an experimenter wishes to search for a 10-state solution. He begins by writing "10" in the space provided:

# of states	10
-------------	----

For mnemonic purposes, he will find it convenient to have the states represented by alphabetic characters or other symbols. For example, he might use acronyms: "Q" for the quiescent state; "G" for the general; "F" for the firing state. Thus, after the number of states is selected, FS5 displays an initial alphabetic choice for the state names:

States	A B C D E F G H I J
--------	---------------------

At any time, the experimenter may write over this display to replace these choices by his own.

THE MESSAGE CENTER

If we remove the burden of tedious work only to replace it with a large set of system rules and procedures to be learned, the experimenter has gained very little. To avoid this pitfall, FS5 has a MESSAGE CENTER which prompts the user on system usage, informs him of conditions, and suggests actions to take when errors occur. In other words, FS5 supplies copious run-time diagnostics.

For example, when the experimenter begins to write in a value for the number of states, FS5 prompts him with

ENTER NUMBER OF STATES
SWEEP TO EXIT.

If he begins to rewrite the external state names, he sees

ENTER NAMES OF STATES
1=QUIESCENT 2=GENERAL
LAST=FIRING
SWEEP PEN TO EXIT.

Furthermore, FS5 guards against such illegal procedures as trying to enter one of the three reserve state names--"#", ".", "?"--in this case by refusing to accept them.

The policy on a user error is to announce it, correct it and leave the system in a usable condition whenever possible, or else inhibit further action until the user makes a correction, and advise him how to do so.

ENTRY, STORAGE AND RETRIEVAL OF FUNCTION VALUES

For a 10-state solution, as many as 891 function values might be needed. As a complication, a large number of productions might be undefined at any given time. FS5 provides several ways to enter, retrieve and alter productions, and takes appropriate action when an undefined production is referenced.

To illustrate, the experimenter may enter a production by writing

DEFINE FROZEN

QQG → G

If he later wishes to recall this value, he writes

DEFINE FROZEN

QQG → ?

and FS5 replaces the "?" by the value; here "G", or by "." if the production is undefined.

Alternatively, the system might have been designed to display the entire state table upon request. However, at any one moment the experimenter is usually interested in only one production. Moreover, many table entries might never be of interest, because no simulation needs them.

SIMPLE SIMULATION OF A FIRING SQUAD

After defining several productions, the experimenter will want to test the function on firing squads of various lengths; FS5 offers several modes of simulation and testing. For a simple case, suppose that a simulation is desired for length 4. The user enters the "4" with the stylus:

LENGTH

4

FS5 responds by initializing the firing squad:

0	Q	Q	Q	G
1	.	.	.	.
2	.	.	.	.
3	.	.	.	.
4	.	.	.	.
5	.	.	.	.
6	F	F	F	F

In addition, the system always provides two productions:

$$\#QQ \rightarrow Q \quad \text{and} \quad QQQ \rightarrow Q$$

where "#" represents the end-marker state. These are the two productions required by the problem statement.

Let us further suppose that the user has entered

$$QG\# \rightarrow G, \quad QQG \rightarrow G, \quad QGQ \rightarrow Q, \quad \text{and} \quad \#GQ \rightarrow G.$$

He starts the simulation by touching

START SQUAD

Then the message center will display

FIRING IN PROGRESS.

Simulation proceeds from time 1 down, left to right on successive rows. Because the Q G G production is undefined, simulation will cease at time 2.

0	Q	Q	Q	G	
1	Q	Q	G	G	
2	Q	G	.	.	Undefined.
3	.	.	.	.	
4	.	.	.	.	
5	.	.	.	.	
6	F	F	F	F	

The message center will contain

ERROR: FUNC NULL & SQAD FREE,

and the undefined production will be displayed,

DEFINE FROZEN

QGG + .

ready for the experimenter to enter a value.

A simulation may be temporarily halted at any time to check its progress. During these manual stops, FS5 continues to advise on system status; and messages are also provided for automatic stops.

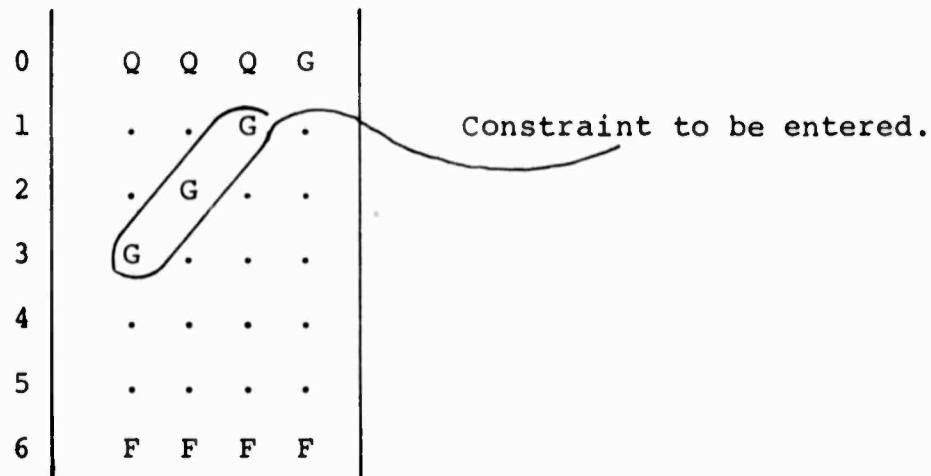
ENTER CONSTRAINTS

With these simple services at his disposal, the experimenter can turn his attention to finding good solution approaches. FS5 enables him to enter two-dimensional skeleton plans which really are a set of constraints on the function behavior.

The state at time zero and the constraints at the firing time are fixed by the problem statement and provided by the system. To enter other constraints, first the user touches

DEFINE CONSTRAINT ,

after which FS5 replies with instructions. Next, the user touches two points to define a line segment on the simulation display, and then a name in the STATE display.



In other words, a constraint is a line segment of states which is "drawn" on the display.

Any number of constraints may be entered, and one may be drawn over another. The "."'s in the display are intended as guides in determining straight lines, and FS5 automatically provides other temporary guides and markers. If an error is made or a change desired, the last constraint entered may be erased:

REMOVE LAST

The ability to enter constraints becomes a powerful tool when used in conjunction with the simulation modes described in the following two sections.

SIMULATION WITH CONSTRAINTS

If the experimenter starts a simulation for length 4 with all productions undefined except for $\#QQ \rightarrow Q$ and $QQQ \rightarrow Q$, and with the three-position constraint of the previous example, then FS5 will define the production $QQG \rightarrow G$ from the constraint. However, the simulation will terminate as shown below because neither is $QG\#$ defined nor is the simulation constrained at the position where $QG\#$ is first required.

0	Q	Q	Q	G	Defined by constraint
1	Q	Q	G	.	Undefined
2	.	G	.	.	
3	G	.	.	.	
4	.	.	.	.	
5	.	.	.	.	
6	F	F	F	F	

The ability to draw large numbers of complicated constraints thus relieves the experimenter of the task of tailoring many individual productions to produce the same behavior; all the necessary definitions are made by the system.

The system also detects contradictions between constraints and previously defined functions. Such an error would have occurred had the resultant of QQG been set to Q. These contradictions often escape notice when simulations are performed by hand.

As an alternative to drawing constraints, a language to describe them might be devised. However, it is hard to imagine a language as easy or as natural to use as the FS5 method.

SIMULATION WITH BACKTRACKING

As mentioned above, exhaustive search for a function might become feasible when relatively few productions remain undefined. The use of constraints also can make exhaustive search feasible, because these constraints act as

implicit definitions. To take advantage of constraints, FS5 was equipped with a widely used method of efficient search called the "backtrack" technique [8-10]. For readers not familiar with backtracking, or who may know it by another name, a brief review is in order.

Many combinatorial problems can be stated in the form, "Find a vector $(s_1, s_2, \dots, s_m)$ which satisfies p_m ," where $s_1, s_2, \dots, s_m$ are to be chosen from a finite set of N distinct objects, and p_m is some property. The "brute force" approach is to form in turn each of the N^m possible vectors, testing whether or not it satisfies p_m . A backtrack algorithm is designed to yield the same result with far fewer trials.

The backtrack method consists of defining properties p_k for $1 \leq k \leq m$ in such a way that whenever $(s_1, s_2, \dots, s_m)$ satisfies p_m , then $(s_1, \dots, s_k)$ necessarily satisfies p_k . The computer is programmed to consider only those partial solutions $(s_1, \dots, s_k)$ which satisfy p_k ; if p_k is not satisfied, then the N^{m-k} vectors $(s_1, \dots, s_k, s_{k+1}, \dots, s_m)$ are not examined by the program. When all choices for s_k are exhausted, the program backtracks to make a new choice for s_{k-1} . If the properties p_k can be chosen in an efficient way, comparatively few cases are considered.

In the firing squad problem, the vector $(s_1, s_2, \dots, s_m)$ consists of production definitions. The backtrack method applied in FS5 serially defines the productions as they are needed in the simulation of a firing squad of fixed length M .

The method begins with all productions undefined except the two required by the problem statement. After initializing the firing squad for length M, the program begins to find the new state of each position in the simulation according to the productions which are already defined. If a production is encountered which is not already defined, and this occurs at an unconstrained position, then the resultant is set to either the firing state or another state, depending on whether or not this occurs at firing time. If the position is constrained, the resultant is set to the constraint value.

The process of serial definition continues until an error occurs. An error is defined to be either a soldier going into the firing state before firing time, a soldier not firing at firing time, or a conflict between a constraint and a production already defined. When an error occurs, FS5 backtracks to find the most recently defined production whose resultant is not the firing state, which is first used where there is no constraint, and for which all the choices of a resultant have not been exhausted. All productions defined after this are now undefined, and this production is set equal to a value which has not yet been tried for it. The program then returns to the position in the firing squad simulation where this production was first defined, and simulation continues from there.

The above process of finding the new state of a soldier and defining production as needed is continued until either

a solution is found for length M or else no productions remain which are alterable. In the latter case, we have tried all possibilities which could lead to a solution for the given length with the given constraints and a given number of states. Thus there is no solution in this form.

The experimenter can request FS5 to simulate in "AUTO" mode, in which case backtracking will be applied to any undefined productions which are needed. Backtrack mode may be used with or without either constraints or explicit production definitions having been entered. Simulation will only cease if either a successful function is found or all possibilities are exhausted.

FROZEN AND FREE PRODUCTIONS

The experimenter can freeze the value of a production if he wishes to prevent its alteration without his explicit consent; the key productions are of this nature. Frozen productions are not altered by any simulation mode. Hence, a frozen production is another form of constraint and, if used, may further reduce backtracking effort. Other productions are termed free because the backtracking mechanism is free to alter them.

SNAP VIEW AND BRIGHT POSITIONS

While in backtracking mode, it is useful and necessary to view the progress of the simulation. Sometimes the experimenter can notice an area where much backtracking occurs,

and enter explicit frozen production or additional constraints to eliminate such bottlenecks. Furthermore, if the constraints are neither numerous nor strong, the number of search possibilities could still be astronomical. In this case, if the experimenter periodically views the progress of the simulation, he can decide when it should be aborted.

With the "SNAP VIEW" option "ON," redisplay of the simulation occurs after each row is completed and also whenever the system must backtrack. Otherwise (and in all cases of "STOP" mode), redisplay occurs only when simulation terminates. Since a position in a simulation at which a production is first used is of special interest, all such positions may be brightened by pushing a button:

BRIGHT

Both features are optional because frequent redisplay significantly increases running time.

HISTORY SCROLL, FREEZING AND DELETION

Although the experimenter may never be interested in seeing the entire production table at one time, he may have occasion to view significant portions of it. A scroll display gives him the list of productions used in the current terminated simulation, in order of original usage, and indicates which are frozen.

If production definitions were generated by constraints or backtracking, he might want to freeze some or discard others. Either can be done by pushing the appropriate button

FREEZE VALUE
DELETE VALUE

and touching productions on the scroll.

IMAGE SOLUTIONS

Experience with the problem, and general consideration of the form that any solution must take, led to giving FS5 another heuristic tool, which requires explanation because its motivation is less obvious than that of other program features.

In any solution, signals must travel the entire length of a squad in both directions because the general, before he can fire, must know that the order to fire has reached the last soldier on the opposite end of the squad. If the signal sent by the general is 1, and the signal returned by the last soldier is 2, then we may think of signal 2 as being the image produced by the reflection of signal 1 from the end of the squad. In other words, the general bounces signal 1 off the end of the squad; the image--echo--returns to him as signal 2.

Experience with various solution methods has demonstrated many other instances in which the image analogy is helpful. For example, suppose that we are applying the technique of successive subdivision, and have contrived a partial skeleton plan:

0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

The general emits signal 1, and it travels to the left at the maximum possible rate of one man per unit of time. This signal arrives at the end of the squad and produces an image, signal 2, which travels at the same rate in the opposite direction.

The general also emits signal 3, and it travels at one-third the rate of signal 1. Thus, signals 2 and 3 meet at the midpoint of the squad and produce the first division point. This central soldier is then promoted to general, and the process can be repeated for each of the two halves.

To repeat the process, the central general sends signal 1 to the left as before, but now a signal 4 is also sent to the right. Signal 4 is intended to behave in the same manner as 1, except that 4 travels in the opposite direction. Signal 4 is, therefore, an image of signal 1, created by reflection about the center of the squad.

Images imply that certain symmetries will probably exist between sets of productions and between pairs of states. Therefore, an additional heuristic for the problem is to look for solutions having the property that for every production $LMR \rightarrow S$ there exists a production.

$$\text{Image (R) Image (M) Image (L) } \rightarrow \text{Image (S)}$$

where the Image function maps the set of states onto itself such that $\text{Image (Image (S))} = S$ for all states S .

In FS5, if the image-solution mode is selected and the user defines a proper image mapping, then whenever a production is defined, the image production is also defined. The image method may be used separately or in combination with constraints and backtracking. Obviously, the image

method also improves the feasibility of exhaustive search because the number of free productions is again reduced.

MISCELLANEOUS

Other controls allow for reinitializations, for simulation testing over any range of lengths up to 500 men, and for hard copy of displays and tables.

V. IMPLEMENTATION EXPERIENCE AND PREREQUISITES

Any system like FS5 endeavors to provide the researcher with tools and response time that encourage and allow him to apply methods of solution which might otherwise be impractical. On the other hand, if the labor of writing the software is greater than the hand calculation it eliminates, a researcher finds small encouragement. In general, if the cost of building an interactive system exceeds the importance of the problem area, the system will not be built. Our feeling is that the cost of FS5 is reasonable, and that costs relative to more important problems will be significantly lower.

The required hardware includes a digital computer, a CRT display with appropriate graphic input device, and associated interface equipment. The choice of input device is crucial to human reaction. A light pen is at best a clumsy pointing instrument, and a typewriter keyboard with display cursor is an unnatural tool. Had these been the only devices available, many FS5 features would have been neither conceived nor implemented. An appliance used in the manner of a pencil, such as the RAND Tablet, is central to the efficacy of interactive problem-solving systems.

FS5 required three software types, exclusive of programming language and operating system: a graphic software system (IGS); routines to service displays and controls; and routines providing non-graphical aids. IGS allows the

user to think globally about displays for his problem, rather than about intricate hardware and bit patterns. Routines to generate and manage displays consist primarily of calls to IGS. Non-graphical routines, such as table maintenance and backtracking, were no different than they would have been if all output was printed.

Thus, the major efforts in writing FS5 were to design displays and to interface with the existing graphic software. With such high-level languages as PL/I or FORTRAN, and a good package such as IGS, this is not a very difficult task.

VI. CONCLUSION

An on-line, graphical, man/machine interactive computer system can provide greatly increased research power over a system lacking these attributes. This is true even when a problem is not inherently graphical. Anyone who is planning a computer system to investigate a difficult problem area should consider extending the design to make it graphical and interactive. Since most medium and large computer facilities already have the necessary hardware and basic software, and since construction of routines to generate and to manage displays is quite simple, the added cost should be very small compared to the extra utility gained.

REFERENCES

1. Moore, E. F., Sequential Machines, Selected Papers, Addison-Wesley, 1964.
2. Goto, Eiichi, "A Minimum Time Solution of the Firing Squad Problem" (Dittoed Course Notes for Applied Mathematics 298, Harvard University), May 1962, pp. 52-59.
3. Waksman, Abraham, "An Optimal Solution to the Firing Squad Synchronization Problem," Information and Control, Vol. 9, No. 1, February 1966, pp. 66-78.
4. Fischer, Patrick C., "Generation of Primes by a One-Dimensional Real-Time Iterative Array," Journal of the Association for Computing Machinery, Vol. 12, No. 3, July 1965, pp. 388-394.
5. Balzer, Robert M., Studies Concerning Minimal Time Solutions to the Firing Squad Synchronization Problem, ARPA SD-146 (Ph.D. Thesis), Center for the Study of Information Processing, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, 1966.
6. Davis, M. R., and T. O. Ellis, "The RAND Tablet: A Man-Machine Graphical Communication Device," AFIPS Conference Proceedings (1964 FJCC), Vol. 26, Part I, Spartan Books, Inc., Baltimore, Maryland, 1964, pp. 325-331; also, The RAND Corporation, RM-4122-ARPA, August 1964.
7. Groner, Gabriel F., Real-Time Recognition of Hand-printed Text, The RAND Corporation, RM-5016-ARPA, October 1966.
8. Walker, R. J., "An Enumerative Technique for a Class of Combinatorial Problems," AMS Proc. Symp. Appl. Math., 10, 1960, pp. 91-94.
9. Golomb, Solomon W., and Leonard D. Baumert, "Backtrack Programming," Journal of the Association for Computing Machinery, Vol. 12, No. 4, October 1965, pp. 516-524.
10. Hall, Marshall, Jr., and D. E. Knuth, "Combinatorial Analysis and Computers," The American Mathematical Monthly, Vol. 72, No. 2, Part II, February 1965, pp. 21-28.